An Improvement of Tree-Rule Firewall for a Large Network: Supporting Large Rule Size and Low Delay

Thawatchai Chomsiri[1], Xiangjian He[2,*], Priyadarsi Nanda[2], Zhiyuan Tan[3]

[1]*Research Center of Information Technology for the Future (ITF),*
Department of Information Technology, Faculty of Informatics
Mahasarakham University, Thailand

[2]*Center for Innovation in IT Services and Applications (iNEXT),*
School of Computing and Communications, Faculty of Engineering and Information Technology
University of Technology Sydney, Australia

[3] *Services, Cyber security and Safety Research Group, Faculty of Electrical Engineering, Mathematics and Computer Science,*
University of Twente, Netherlands

[*]Contact: Xiangjian.He@uts.edu.au

***Abstract*— Firewalls are important network devices which provide first hand defense against network threat. This level of defense is depended on firewall rules. Traditional firewalls, i.e., Cisco ACL, IPTABLES, Check Point and Juniper NetScreen firewall use listed rule to regulate packet flows. However, the listed rules may lead to rule conflictions which make the firewall to be less secure or even slowdown in performance. Based on our previous research works, we proposed the Tree-Rule firewall which does not encounter such rule conflicts within its rule set and operates faster than the traditional firewalls. However, in big or complex networks, the Tree-Rule firewall still may face two main problems. 1. Firewall administrators may face difficulty to write big and complex rule. 2. Difficulty to select appropriate attribute column for the Root node.**

In this paper, we propose an improved model for the Tree-Rule firewall by extending our previous models. We offer the use of combination between IN and OUT interfaces of the firewall to separate a big rule to many small independent rules. Each separated rule then can be managed in an individual screen. Sequence of verifying attributes, i.e., Source IP, Destination IP and Destination Port numbers, can be ordered independently in each separated rule. We implement the two main schemes on Linux Cent OS 6.3. We found that the improved Tree-Rule firewall can be managed easily with low processing delay.

Keywords*— *Firewall, Tree-Rule firewall, Network Security, Large Rule Size, Low Delay

I. Introduction

The firewalls were invented since 1990s [1] and have been developed to operate more secure and faster. From the first era of the firewalls until today, they still regulate packet based on a listed rule. The listed rule is the set of rule sequence which consists of a condition and action. If incoming packets' information, i.e., Source IP, Destination IP and Destination Port, are matched with the condition, the packets will be accepted else, denied followed by an action specified in the rule. In the listed rule set of traditional firewall, there may be shadowed rules [2] or redundant rules which can make firewall operate slower because the firewall will waste its operational time to verify against these rules. Moreover, shadowed rules can cause security problems because protection rules can be shadowed by other rules above. These problems of traditional firewalls have been identified and published in our previous research [3]. In [4], we proposed the new type of firewall called the 'Tree-Rule firewall', and proved that it can offer less rule conflict and can operate faster than the traditional firewall. However, the first version of Tree-Rule firewall [4] works as a packet filtering firewall not a stateful firewall. Consequently, we then proposed a stateful mechanism [5] providing more security for the networks. However, for a large network which consists of many servers, opened ports, user groups, and network branches, the Tree-Rule firewall requires a big rule too. Therefore, in this paper, we will propose solutions for these problems. We firstly introduce background, previous works, and problems in Section II. We then explain the details of our approach in Section III. In Section IV, we provide implementation of our proposed scheme and conduct several experiments. Finally, we conclude this paper in Section V along with future directions for our research.

II. Background, Previous works and Problems

A. *Background and Previous Works*

Many researchers have studied conflicts within rule list of the traditional firewall (listed-rule firewall). Occurrence of shadowed rules and redundant rules have been presented in [2][6]. Many research findings presented in [2][6][7] suggest schemes to remove these bad rules. The Binary Decision Diagrams (BDDs) [8], Constraint Logic Programming (CLP) [9], and Fireman Toolkit [10] analyse and get rid of rule conflicts within rule set in listed-rule firewall. The methodology in [11] proposed to use 'jump' command in IPTABLES to decrease rule conflicts and increase IPTABLES firewall speed. These studies are evidences indicating that the traditional firewalls have many problems because they depend on the listed rule. We have studied and found a real cause of these problems, and suggested an idea to avoid the problems by using the Tree-Rule firewall [3] [4].

The Tree-Rule firewall not only use tree structure for its rule but it also decide packets based on such tree structure too. The tree structure of rule in a user's view provide an ease of design without rule conflict. This has been proved in [3][4] already. Also, the tree structure of rule provides fast packet processing decision because its big order "O" of packet decision time consumption is in the logarithm term (see [4]). However, in the rule designing aspect, we found that both listed-rule firewall and Tree-Rule firewall have problems with large rule size.

B. *Problems of Tree-rule firewall on a large network*

We define large network is a network which has many servers, many user groups (i.e. normal users, database users, and admin users), many network zones or branch offices placed distributedly as presented in Fig 1 below. Each server opens many service ports, and local users can access to internet using those ports. As the size of network grows, the rule tree within Tree-rule firewall also grows accordingly.

1) Big Rule Tree: The growth of rule tree will not be increased in a horizontal direction (width direction) because a number of attribute (header column for rule tree, i.e., Source IP address, Destination IP address and Destination Port) was fixed by firewall administrators / designers since the first node of the rule tree has been created. Therefore, the rule tree will grow vertically; for example, the growth of rule tree can cause an increase number of line within the Root node as presented in Fig 2, while number of nodes in second and third column increases as well. In the previous version of our Tree-Rule firewall, firewall editor (GUI software) shows a scroll bar in the right hand side of design screen if height of rule tree becomes greater than height of the screen. Firewall administrators (firewall rule designer) can drag the scroll bar up and down to modify or manage the rule. However, as the rule size becomes very large, firewall administrators have to scroll up and down many times to see across a page. This can bring inconvenience to firewall administrators so much.

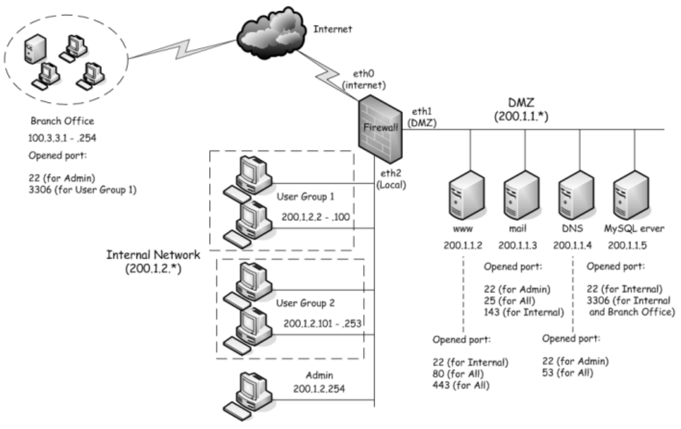

Fig 1. An example network

Fig 1 presents an example of medium size network which correspond to the rule tree shown in Fig 2. The rule tree in Fig 2 can be displayed in one screen. However, if the complexity or size of network is increased, the rule will be bigger than the screen.

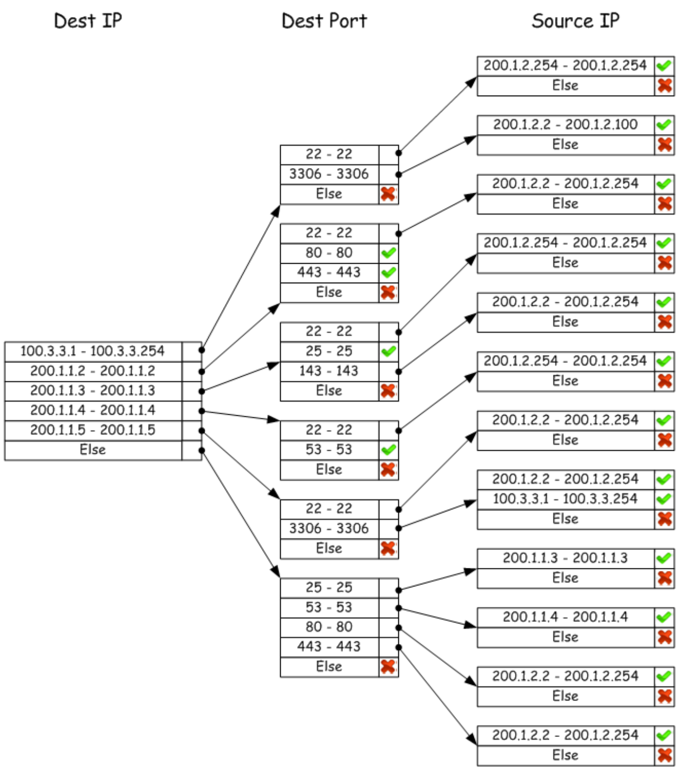

Fig 2. Rule tree corresponding to the network in Fig 1

Note: We will use an example network in Fig 1 and rule tree in Fig 2 for demonstrating an optimization methods and its details are presented in the next Sections.

2) Column ordering: As we mentioned in the previous sub section, the rule size is bigger than the screen is not the only one problem but another problem is that the large network can cause difficulty for selecting an appropriate attribute for the Root node (including other rest columns). First of all, we would like to give details about selecting an appropriate attribute for the Root node.

Selecting an appropriate attribute for the Root node (first column) can help us to design firewall rule easily. For example, in a network which emphasizes to provide servers' services, we should select the Destination IP Address to be the Root Node (see Fig 2) because it is easy to imagine that "What servers we have got?" (e.g. 'Dest IP' column in Fig 2), "What ports each server has to open?" (e.g. 'Dest Port' column in Fig 2), and "Who can access these ports?" (e.g. 'Source IP' column in Fig 2). For another example, in the network which consists of many user groups and focuses on a concept what ports on the internet each user group can access to, it will be fine if we choose Source IP address to be the Root node. In this case, the second column should be Destination Port. This is because we can imagine the local users as source IP addresses, and then imagine ports which they want to go (i.e., port 80 for connecting to web servers on the internet).

In some cases that mix between two cases as we explained above, for example, the network which has many group of users and has many kinds of server in DMZ. It is difficult for firewall administrators to select an appropriate attribute for the Root node. In the next section, we will propose the solution for this case. We also propose a solution for the big rule size as well.

III. Our Approach

We solve the two main problems which we mentioned in Section II by using the following methods:

a. Using combination of ‘in’ and ‘out’ interfaces on separate design pages

b. Using independent column ordering for each separate design page

However, before explaining these schemes, we will adjust (optimize) the rule tree first because this optimization can improve the rule tree to be more compact. We will optimize the rules by using 'single action', 'single value', and 'IP address & port name'.

Using a single action is the method that the rule designers must create rule and for only one action. The action can be only Accept, or can be only Deny presented in Fig 3.

Firewall will have a 'default policy' which means that if any incoming packet cannot be matched with the rule, it will be decided by the default policy. Examples of a default policy are 'Implicit Deny' of Cisco ACL (Cisco's Implicit Deny has been set to be Deny permanently and cannot be changed), IPTABLES' default policy (we can change its default policy to be Accept or Deny). In the case of the Tree-Rule firewall, firewall designers can set the default policy to be Accept or Deny; however, we recommend Deny because it will then make the network more secure without allowing unknown packets. With the 'single action' method, an action for each rule path will be same, and rule tree will be slightly decreased (see Fig 3 compare with Fig 2).

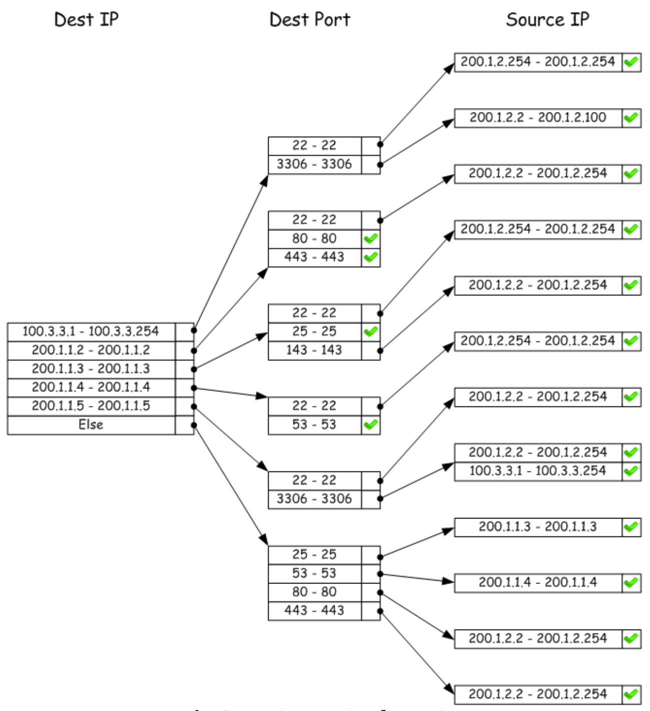

Fig 3. Using a single action

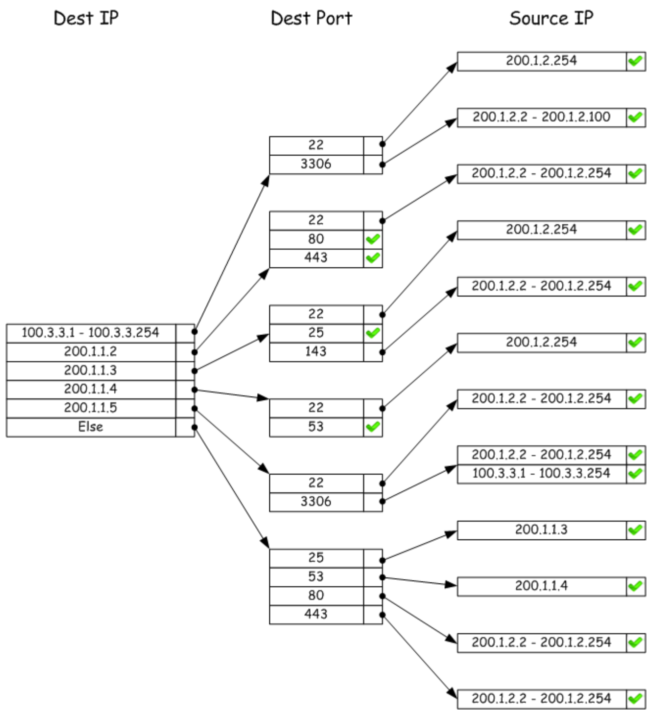

Fig 4. Using a single value

Using a single value will allow firewall rule designers to input a single IP address or a single port number in the line within a node instead of specifying IP address range or port range. This method allows a single number in case that the starting number and the ending number are same, i.e., specify '200.1.1.2' instead of '200.1.1.2-200.1.1.2' as can be seen in Fig 4. With this method, number of character within rule tree will be decreased and rule designer can understand the rule easier. However, in firewall's memory, a data structure corresponding to the rule still will be in the range of number while the GUI can interact with designers using a single number.

Using an IP address name and a port name is the method that allows naming for the IP address (es) and port (s) in order to add them into the rule tree. For example, we can define the name 'Web Server' as an IP address number 200.1.1.2, and we can define the name 'UserGroup1' as the IP address range 200.1.2.2 - 200.1.2.100 (see Fig 5).

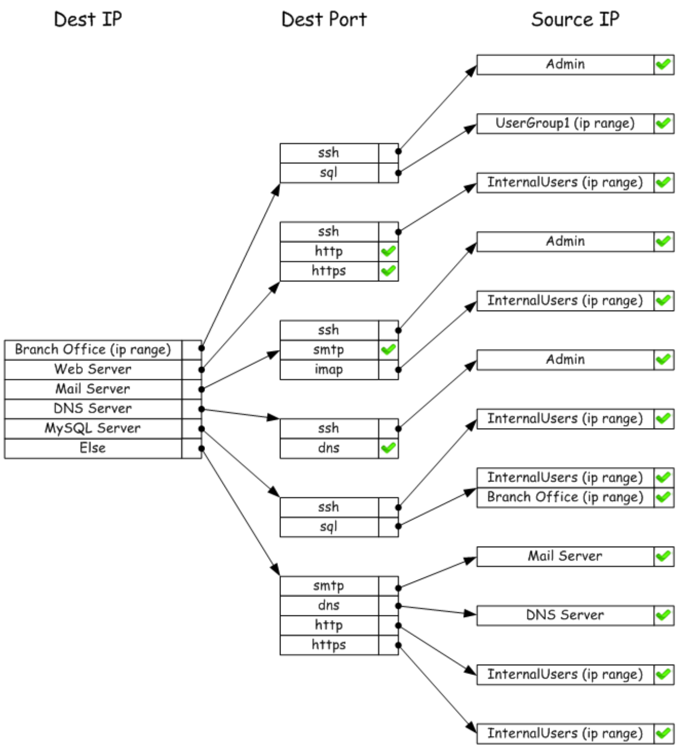

Fig 5. Using an IP address & port name

Using an IP address name and a port name will provide easy way for firewall rule designers to understand the rule. The GUI of firewall rule editor can display the rule in both modes (number and name) up to firewall rule designers. They can switch to the 'name mode' and switch back to the 'number mode' by clicking on a menu or using short keys easily to make sure that the rule they design is correct.

Many such methods similar to the ones explained above can minimize a rule size and help rule designers to manage rule tree easily. However, these methods are not the highlight of this paper. Apart from this, we propose a scheme to reduce the rule size significantly which is the main focus of our paper. Our proposed scheme also gets rid of the problem about selecting appropriate attribute for the Root node too.

A. *Using combination of 'in' and 'out' interfaces*

We can divide an original big rule tree to many small rule trees working independently from each other. This means that we will get many screens of rules, and these rules will have no conflict with other rules in other screens. This separation can be done using combination between 'in' and 'out' interfaces. For example, the network in Fig 1 which has three network interface (eth0, eth1, and eth2) can combine its pair of two network interface cards together and can be presented as:

eth0 -> eth1

eth0 -> eth2

eth1 -> eth0

eth1 -> eth2

eth2 -> eth0

eth2 -> eth1

The 'ethX -> ethY' notation is used to indicate incoming packets to the firewall using ethX, and going out from the firewall using ethY. In this example, we can call the 'eth0 -> eth1' combination as the 'Internet -> DMZ' combination because they are of same meaning (see Fig 1). Other cases can be explained in a similar ways. We use a Tab control in our GUI firewall rule editor to separate a design screens as shown in Fig 6, Fig 7, Fig 8 and Fig 9.

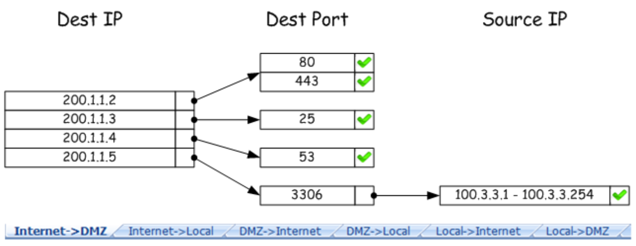

Fig 6. 'Internet -> DMZ' separated design screen

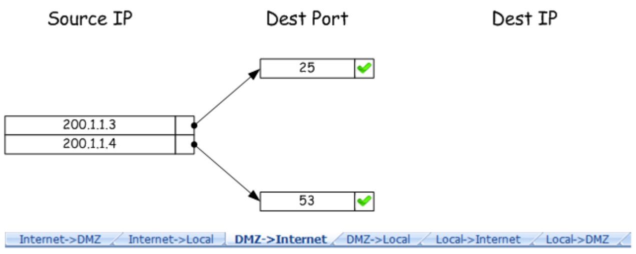

Fig 7. 'DMZ -> Internet' separated design screen

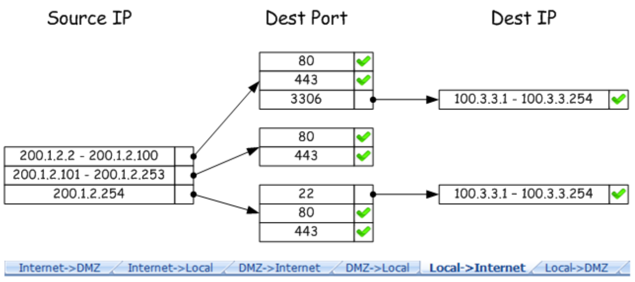

Fig 8. 'Local -> Internet' separated design screen

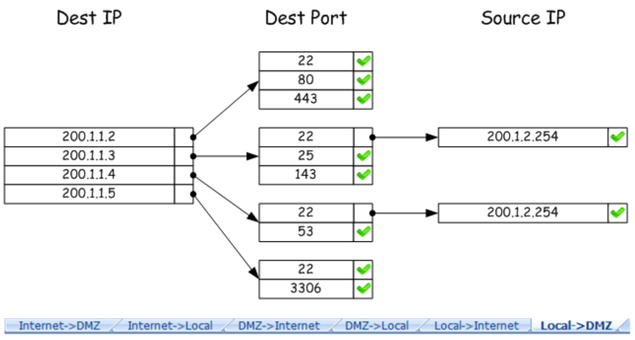

Fig 9. 'Local -> DMZ' separated design screen

The Fig 6 is a design screen of the 'eth0 -> eth1' combination. The rule displayed on Fig 6 will be corresponding to policy which will regulate packets originated from internet and destined to DMZ. If we consider the rule in Fig 4 or Fig 2, and compare with rules in Fig 6, 7, 8, and 9, we can see that the rule in separated design screen (Fig 6, 7, 8, and 9) are obviously smaller than the original rule. For instance, the rule in Fig 6 (Internet -> DMZ) will have only line and node which are corresponding to packet direction. As we can see the Root node ('Dest IP' column) of Fig 6, the lines inside Root node represent only IP addresses of servers situated in DMZ. Also, we can see that a number of nodes within the 'Source IP' column (Fig 6) has been reduced significantly. This is because the packets corresponding to the 'eth0 -> eth1' must be originated from internet.

Some separated rules will have very small size such as the rule in Fig 7 which is used to regulate packets originating from the DMZ and destined to internet. It can be possible that some separated design screens have no rules. This is because packets travelling in some directions cannot be allowed, such as packets in the Internet->Local (eth0->eth2) direction. This prevention can protect users' computers in the Local Network from hackers in internet.

Therefore, with this scheme (combination between ‘in’ and ‘out’ interfaces), a big rule can be divided into several small rules which provide an easy way for firewall rule designer to manage firewall rules.

B. *Using independent column ordering for each separate design page*

According to the problem we mentioned in Section II on difficulty to select an appropriate attribute for the Root node, we can solve the problem by providing the new version of rule editor GUI which allows firewall rule designer to determine a sequence of attribute columns in each separated design screen independently. For example, firewall rule designer set the sequence of attribute column in Fig 6 to be 'Dest IP', 'Dest Port', and 'Source IP' respectively while the sequence in Fig 7 was differently set to be 'Source IP', 'Dest Port' and 'Dest IP' respectively.

In fact, by analysing this scheme, we can find that the sequence of attribute columns can be adjusted because of combination between ‘in’ and ‘out’ interfaces and separating a rule.

IV. Implementation and Experimentation

Similar to our previous schemes [4] [5], we implement the proposed schemes based on the Netfilter [12][13][14] module. We hook packets' events using a technique presented in [15] by calling the 'nf_register_hook(&nfho);' function. Before calling this function the hooking function must be declared first, such as in the line: 'nfho.hook = hook_func;'. When packets arrive at the firewall, the 'hook_func' will be called. It will receive several important parameters as shown below:

```
unsigned int hook_func(unsigned int hooknum,
        struct sk_buff *skb,
        const struct net_device *in,
        const struct net_device *out,
        int (*okfn)(struct sk_buff *))
{
}
```

An implementation of the Tree-Rule firewall in [4] and [5] use only Source IP address, Destination IP address, Source port and Destination port from the 'struct sk_buff *skb'. In this paper, we verify more information, i.e., the IN interface and the OUT interface. With this function, we read them from the 'const struct net_device *in' and 'const struct net_device *out'.

We create the Tree-Rule firewall using C on Cent OS 6.3 Linux. It operates as a kernel module and runs in a kernel level. That is, our original firewall source code, firewall.c, will be compiled to the firewall.ko and will be executed by the command '# insmod firewall.ko'. We develop rule editor GUI using C# on Windows. The firewall rule will be created by GUI and will be sent to the core firewall running on Linux. The rule structure will be slightly modified for appendding information about a sequences of attribute columns, and network interface card combinations.

A stateful mechanism presented in [5] will be modified in few parts; for example, for calculating the entry of hashing table (the parameter 'Entry' in [5]), we cancel the use of Max and Min [5] function because the new schemes can provide information of packets directions already. Thus, we will change the formula from:

Entry = Hash(Max(Source _IP, Destination_IP), Min(Source_IP, Destination_IP), Max(Source_Port, Destination_Port), Min(Source_Port, Destination_Port), Protocol_Type, Key).

to

Entry = Hash(Source _IP, Destination_IP, Source_Port, Destination_Port, In_Interface, Out_Interface, Protocol_Type, Key).

In [5], we used Max and Min function for this formula because the firewall did not know packet directions. In contrast, in this paper, the packet directions come along with the parameter '*in' and '*out'. These parameters will be the input parameters for a new formula to calculate the 'Entry' (see [5]).

In Section III, we can see that separating a rule can bring several small rules. However, it requires an additional task on the firewall CPU because firewall have to check the IN interface and the OUT interface while it may work faster with a new small rules corresponding to these interfaces. Thus, we have to experiment and study an operational speed of the proposed model compared to the previous one; especially, in the large network.

We create a big rule directly by creating a special code (C language) within firewall kernel program. The big rule for this study will be created from the network information below:

- the network has 8 servers and each server opens 8 ports

- the first 4 servers serve users from internet (all IP address)

- the second 4 servers serve 4 groups of internal users

- the firewall has 3 interfaces which are Internet (eth0), DMZ (eth1) and Local (eth1)

We experiment in 4 cases

Case #1 Attack non-existing IP addresses in DMZ, i.e., IP Scanning and DoS Attack

Case #2 Attack the Web Server on other ports (not port 80)

Case #3 Generate normal packets which SourceIP=internet IP address, DestIP= Web Server, DestPort=80

Case #4 Generate normal packets which SourceIP=Local IP address, DestIP= internet IP address, DestPort=80

We use 'hping' command on BackTrack 5R3 to generate packets by using the command like this:

```
# hping3 192.168.22.2 -a 192.168.11.2 -p 333 -S -s +1 -d 1440 -i u1000
```

We also measure the processing time on packet-decision process. Starting time will be recorded when packet arrive at the hooking function [15]. Ending time will be recorded before the line 'return NF_ACCEPT' and 'return NF_DROP' [5][15]. We experiment on VMware with more than 1,000 packets before taking average for every case. We test on both previous model [4][5] and the proposed model. The previous model will then be compared with the proposed model. The experimental results are presented in Table 1.

Table 1. Time consumption of packet decision on Tree-rule firewall

	Processing Time for one packet (nanosecond)	
	The Previous Model	The Proposed Model
Case #1	194.28	203.79
Case #2	227.41	208.14
Case #3	261.83	216.32
Case #4	286.92	221.86

The experimental result shows that operational speed of the previous model and the proposed model are slightly different. Although the proposed model has to verify against IN and OUT interface, it still gives us a good performance in speed of operation. Especially, in normal case (Case #3 and Case #4), it can provide a better performance. This is because number of line within the Root node in the proposed model is less than that in the previous model. Not only in the Root node, but other nodes (in other columns) are also smaller than the original nodes in the previous model. This is resulted from separating one big rule to many small rules. However, in some cases, the previous model will be faster. For example, in the Case #1, searching servers' IP address in Root node with the key (key is an IP address we want to find) which is greater than Maximum IP address in Root node, or less than Minimum IP address in Root node, the searching operation will be stopped quickly. This is because the key will be compared with the Minimum and Maximum first. If key is less than minimum or greater than maximum, the searching operation will return 'Not found!' and not necessary to operate binary searching. Therefore, in Case #1 where a key is outside the scope for the Root node, second and third column nodes, a decision can be made quickly. The Case #1 of the proposed model may be fast as in the previous model. However, the proposed model should verify against IN and OUT interface which require more time. Thus, the proposed model may be slower at time than the previous model in some cases.

V. Conclusion and Future works

In this paper, we propose two main schemes to resolve problems of Tree-Rule firewall in large network environments. From our study, the problems came from big rule tree which cannot display on one screen and make difficulty to select appropriate attribute column for the Root node. The proposed solutions are (1) using combination between IN and OUT interfaces, and separating one big rule to many small rules, and (2) using independent column ordering for each separate design screen. After deploying these schemes, we found that firewall rule can be easily managed. We also measure decision time interval for the proposed model compared to the previous model. The experimental results show that operational speed of the proposed model still is high (low processing delay) although it should verify against IN and OUT interfaces. In future, we will extend our Tree-Rule firewall to co-operative firewalls suitable for cloud network environment.

References

[1] W. Cheswick, S. Bellovin, A. Rubin, Firewalls and Internet Security: repelling the wily hacker, Addison-Wesley Professional, 2003.

[2] E. Al-Shaer, H. Hamed, Firewall policy advisor for anomaly detection and rule editing, in: Proceedings of the IEEE/IFIP Integrated Management, IM, 2003, pp. 17–30.

[3] T. Chomsiri, X. He, P. Nanda, Limitation of listed-rule firewall and the design of tree-rule firewall, in: Proceedings of the 5th International Conference on Internet and Distributed Computing Systems, China, 2012, pp. 275–287.

[4] X. He, T. Chomsiri, P. Nanda, Z. Tan, Improving cloud network security using the Tree-Rule firewall, Future Generation Computer Systems, Elsevier, 30 (2014) 116-126.

[5] T. Chomsiri, X. He, P. Nanda, Z.Tan, 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom.2014), 2014, pp. 122-129.

[6] C. Pornavalai. T. Chomsiri, Firewall Policy Analyzing by Relational Algebra, In: proceeding of the 2004 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) , 2004, pp. 214-219.

[7] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan, Conflict classification and analysis of distributed firewall policies, IEEE Journal on Selected Areas in Communications 23 (10) (2005) 2069-2084.

[8] S. Hazelhusrt, Algorithms for Analyzing Firewall and Router Access Lists, Technical Report TR-WitsCS-1999, Department of Computer Science, University of the Witwatersrand, 1999.

[9] P. Eronen, J. Zitting, An Expert System for Analyzing Firewall Rules, In: Proceedings of the 6th Nordic Workshop on Secure IT-Systems (NordSec), 2001, pp. 100-107.

[10] L. Yuan, J. Mai, Z. Su, FIREMAN: A toolkit for Firewall modeling and analysis, In: Proceedings of the 2006 IEEE Symposium on Security and Privacy, 2006, pp. 199-213.

[11] L. Zhao, A. Shimae, H. Nagamochi, Linear-tree rule structure for firewall optimization, In: Proceedings of Communications Internet and Information Technology, 2007, pp. 67-72.

[12] R. Rosen, Netfilter, Linux Kernel Networking, Apress, (2014) 247-278.

[13] The netfilter.org project, 2014, http://www.netfilter.org/.

[14] P. Ayuso, Netfilter's Connection Tracking System, LOGIN;, The USENIX magazine, 32 (2006) 34-39.

[15] Fidel, Vidal, and José María. "Mecanismo para el acceso público a servidores con direccionamiento privado." (2011).